第 5 章　面向数据流的分析方法

面向数据流、面向对象、面向数据的分析方法均属需求建模方法。它们都有一组语言机制，供需求分析人员表达用户需求并且构造软件模型。此外，它们还含有一些规则和经验知识，指导分析人员提取需求并使用户需求精确化、全面化、一致化。

面向数据流的分析方法是结构化分析方法族中的一员。它具有明显的结构化特征。结构化分析方法的雏形出现于 20 世纪 60 年代后期。但是，直到 1979 年才由 DeMarco 将其作为一种需求分析方法正式提出，此后它得到了迅速发展和广泛应用。20 世纪 80 年代中后期，Ward 和 Mellor、Hatley 和 Pirbhai 在结构化分析方法中引入了实时系统分析机制，Harel 等人研制了面向复杂实时反应式系统（complex real-time reactive system）的开发环境 STATEMATE，这些扩充使得传统的结构化分析方法重新焕发出生命力。

5.1　结构化分析概述

结构化分析（Structured Analysis，SA）方法就是面向数据流自顶向下、逐步求精进行需求分析的方法。

目前许多计算机系统是用来取代当前已存在的人工数据处理系统的，即用目标系统取代当前系统。那么，如何建立一个目标系统呢？可以按照下述 5 个步骤进行。

（1）理解当前的现实环境，建立当前系统的具体模型。

要理解当前系统是怎么做的，并将现实中的事物用数据流图等形式表达出来。当前系统具体模型示例如图 5-1 所示。

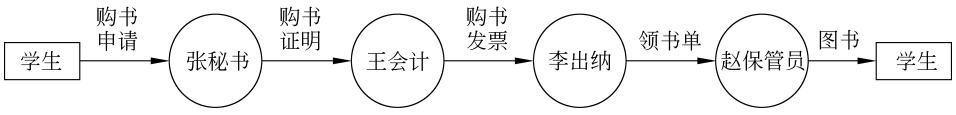


图 5-1　学生购买图书的具体模型

（2）从当前系统的具体模型抽象出当前系统的逻辑模型。

本步骤的目的是去粗取精。即除去具体模型中的非本质因素，获得反映系统本质的逻辑模型。

在本步骤中，可以反复问以下问题：这个加工是否必须这样做？这个文件是否必须这样组织？通过这样的抽象过程，将必需的功能从实现这些功能所采用的方式中分离出来，可去除非本质的因素。当前系统抽象模型示例如图 5-2 所示。

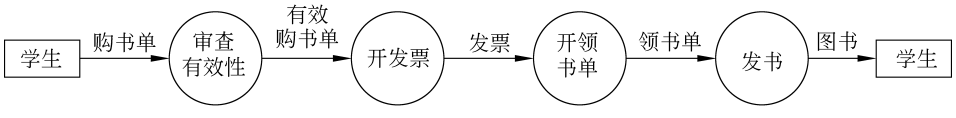


图 5-2　学生购买图书的逻辑模型

(3) 分析目标系统与当前系统逻辑上的差别,建立目标系统的逻辑模型。

本步骤可以分以下3步进行:

① 决定变化的范围。对当前系统的数据流图,从底层开始逐个检查每一个基本加工与目标系统中相应的功能是否一样,若不一样,就属于变化的部分。这样,当前系统的数据流图就被分成不变的部分与变化的部分。只需重新分解变化的部分即可。

② 将变化的部分看成一个加工,将其已确定的输入输出数据流画出。

③ 借助分解技术,由外向内对变化的部分进行分析,创建新系统。

经过上述3步,就获得了目标系统的逻辑模型,如图5-3所示。

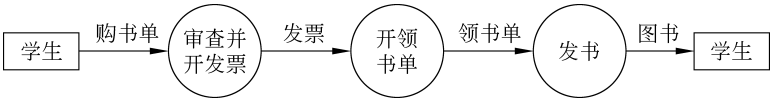


图5-3 计算机售书系统的逻辑模型

(4) 为目标系统的逻辑模型作补充说明。

首先,说明目标系统的人机界面,分为以下两步:

① 确定人机界面。逐个检查目标系统的逻辑模型中的每一个基本加工是由计算机完成的还是由人工完成的,所有由计算机来实现的基本加工都属于目标系统的人机界面。

② 重新绘制数据流图的顶层图,画出软件系统的范围。

其次,说明至今尚未详细考虑的一些细节,主要包括以下4个方面:

① 出错处理。说明在每种出错情况下系统如何处理。

② 系统的启动和结束。说明系统如何开始工作并进入稳定状态,说明系统结束工作的方式。

③ 系统的输入输出格式。

④ 性能方面的要求,如响应时间等。

改进后的目标系统逻辑模型示例如图5-4所示。

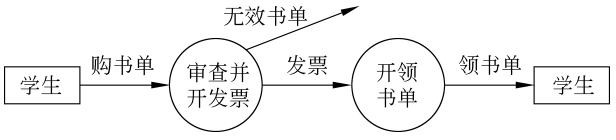


图5-4 改进后的计算机售书系统模型

(5) 对需求说明进行复审,直到确认文档齐全并且符合用户的全部需求为止。

5.2 数据流图

可以认为,一个基于计算机的信息处理系统由数据流和一系列转换构成,这些转换将输入数据流变换为输出数据流。数据流图(Data Flow Diagram,DFD)是软件系统逻辑模型的一种图形表示(graphic representation)。数据流图反映的是客观现实问题的工作过程。它用简单的图形符号分别表示数据流、加工、文件以及数据源点和终点。数据流图中没有任何具体的物理元素,只是描述数据在系统中的流动和处理的情况,具有直观、形象、容易理解的优点。

5.2.1 数据流图的基本成分

数据流图中使用的图形符号如下：

- 数据流用——►表示。
- 加工用◯表示。
- 文件用—或═表示。
- 数据流的源点和终点用□表示。

每个成分都采用适当的名字进行命名。下面先看一个例子，假定要为某培训中心研制一个计算机管理系统，其数据流图如图 5-5 所示。首先需分析这个系统应该做些什么，为此必须分析培训中心的业务活动。培训中心是一个功能复杂的机构，它为有关行业的在职人员开设许多门课程，有兴趣的人可以来电或来函报名选修某门课程。培训中心要收取一定的课程费用，学员通过支票付款。学员也可以来电或来函查询课程计划等有关事宜。培训中心的日常业务是：将学员发来的函电收集分类后，按几种不同情况处理。

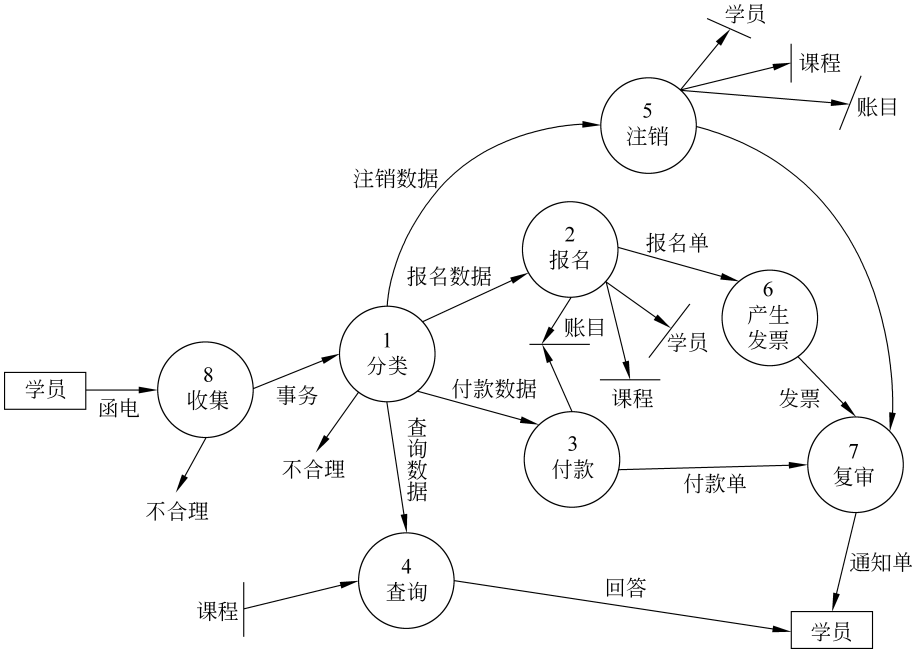


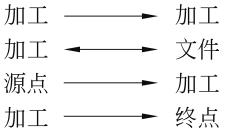
图 5-5 某培训中心数据流图

- 如果是报名的，则将报名数据送给负责报名事务的职员。他们要查询课程文件，检查某课程是否满额，然后在学员文件、课程文件上登记，并开出报名单交财务部门。财务人员再开出发票，经复审后通知学员。
- 如果是查询的，则交查询部门查阅课程文件后给出答复。
- 如果是想注销原来已选修课程的，则由注销管理人员在课程、学员、账目文件上作相应修改，经复审后通知学员。
- 对一些要求不合理的函电，培训中心将相应地处理。

下面结合该例，对数据流图的 4 种基本成分进行说明。

1. 数据流

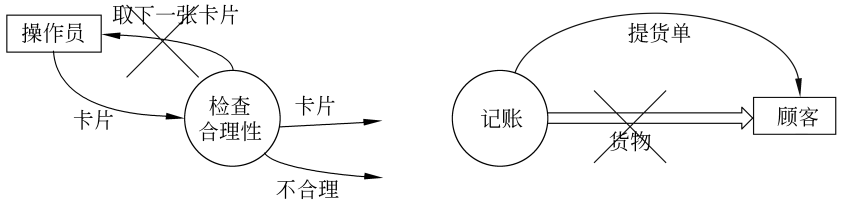
数据流由一组固定成分的数据组成。例如,数据流“报名数据”由“姓名”“年龄”“性别”“单位名”“课程名”等成分组成。数据流反映的是数据的流动方向,而它的流动方向一般有如下几种情况:



注意:两个具体的成分之间可以有几个数据流,但这几个数据流无任何联系且不是同时流出的。

一般从数据流的组成成分或实际含义角度给每个数据流命名。只有流向文件或从文件流出的数据流不必命名,因为这时文件名就可以说明问题了。但是,有如下两点需要特别注意:

(1) 数据流与控制流不同。在控制流上没有任何数据沿着箭头流动,因此在数据流图中不应画出控制流。示例如图 5-6(a)所示。



(a) 数据流与控制流的区别　　(b) 数据流与物流的区别

图 5-6　数据流与控制流、物流的区别

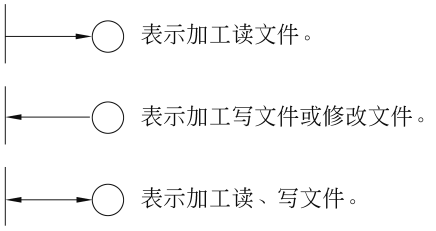
(2) 数据流与物流不同。不能把现实环境中的实物名作为数据流名,软件系统只能处理数据,不能处理实物。示例如图 5-6(b)所示。

2. 加工

加工用于反映对数据进行的某种操作。其命名采用用户习惯使用的且反映加工含义的名字,并加上编号(说明这个加工在层次分解中的位置,见 5.2.3 节)。加工的名字可以是一个动词,也可以由一个具体的及物动词加上一个具体的宾语构成,例如“报名”“产生发票”“查询”等。

3. 文件

文件用于暂时保存数据。它的名字应当适当选择,以便于理解。加工与文件之间的数据流向有如下几种:



4. 源点与终点

为了便于理解，有时可以画出数据流的源点与终点来反映数据的来源与归宿。源点与终点通常是存在于系统之外的人员或组织，例如，图 5-5 中的“学员”是数据流“函电”的源点，也是数据流“通知单”的终点。画出源点和终点只是起到注释作用以帮助理解，由于它们是系统之外的事物，开发人员对它们不是很关心的，所以源点和终点的表达不必很严格。

5.2.2 由外向内画数据流图

在需求分析阶段，只是将现实情况反映出来，而不是急于想象未来的计算机系统是怎样的。所以，分析员可以根据不同的问题，使用不同的方法画数据流图，但原则都是由外向内进行，由外向内是一种比较自然而且有条理的思考过程。具体就是：首先画出系统的输入数据流和输出数据流，然后再考虑系统的内部；每一个加工也是如此，先画其输入和输出，再考虑其内部。

1. 画系统的输入和输出

刚开始进行需求分析时，系统究竟应包括哪些功能还不清楚，所以应该使系统的范围稍大，把可能有关的内容都包括进去。此时应向用户了解系统从外界接收什么数据、系统向外界送出什么数据等，然后根据他们的答复画出数据流图的外围。而上面两个问题的回答分别构成了系统的输入和系统的输出。例如，培训中心管理系统从外界接收的数据是“函电”，向外界送出的数据是“通知单”，则数据流图的外围如图 5-7 所示。

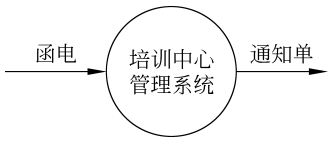


图 5-7 数据流图的外围

2. 画系统内部

此时需逐步将系统的输入和输出数据流用一连串加工连接起来，一般可以从输入端逐步画到输出端，也可以从输出端追溯到输入端。加工应处于数据流的组成或值发生变化的地方。对每一个数据流，应该了解它的组成是什么，这些组成项来自何处，这些组成项如何组合成这一数据流，为实现这一组合还需要什么有关的加工和数据等。另外，数据流图中还要画出有关的文件，即各种存储的数据，此时也应了解文件的组成情况。对培训中心管理系统来说，在数据流图的外围基础上由外向内进行分析，就可画出完整的数据流图。

3. 画加工的内部

如果加工内部还有一些数据流，则可将这个加工分解为几个子加工，并在子加工之间画出这些数据流。先为数据流命名，再为子加工命名。

4. 忽略琐碎的枝节

画数据流图时，应重点关注主要的数据流，暂时不考虑一些例外情况、出错处理等枝节性问题，只表示出这种数据流即可。

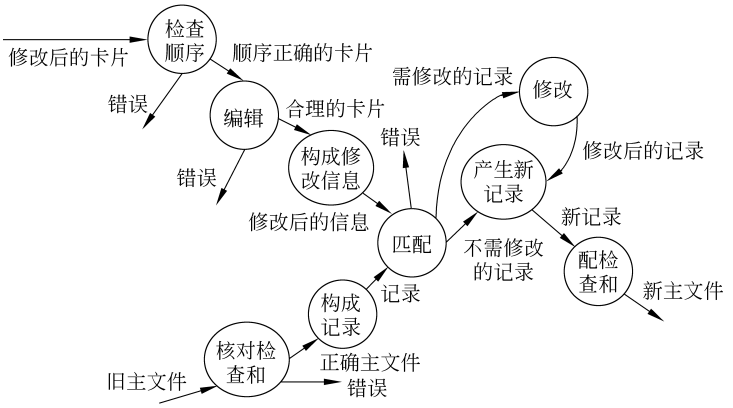
5. 随时准备重画

理解一个问题总要经过从不正确到正确、从不恰当到恰当的过程，一次就成功的可能性是很小的，对复杂的问题尤其如此。分析员应随时准备抛弃旧的数据流图，用更好的版本来代替它。

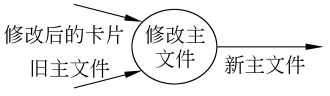
5.2.3 分层数据流图

对于一个系统，特别是一个较大的复杂系统，一次性分解到位一般是不容易的。为了控制复杂性，结构化分析方法采用了分层技术，逐层分解，有控制地逐步增加细节，实现从抽象

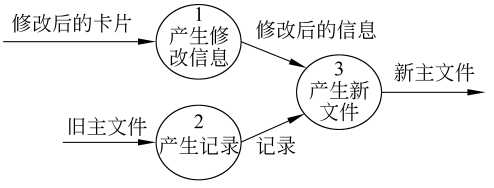
到具体的逐步过渡,这有助于理解复杂问题。用数据流图来描述逐层分解的过程,就得到了一套分层的数据流图,如图 5-8 所示。



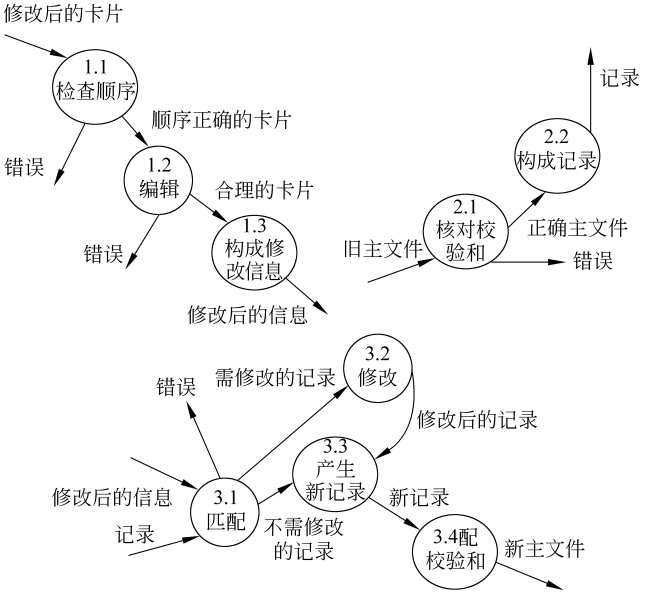
(a) 完整的数据流图



(b) 顶层数据流图



(c) 中间层数据流图



(d) 底层数据流图

图 5-8　分层数据流图的形成过程示例

分层的数据流图由顶层、中间层和底层组成。顶层数据流图说明了系统的边界，即系统的输入和输出数据流，顶层数据流图只有一张。底层数据流图由一些不必再分解的加工组成，这些加工都已足够简单，称为基本加工。在顶层数据流图和底层数据流图之间的是中间层数据流图，它描述了对每个加工的分解，而它的组成部分又要进一步被分解。较小的系统可能没有中间层，而大的系统中间层可达八九层之多。

5.2.4 自顶向下画分层数据流图

本节讨论画分层数据流图中应注意的几个问题。

1. 编号

为了便于管理，需按以下规则为数据流图和其中的加工编号。

(1) 子图的编号(即子图号)就是父图中相应加工的编号。

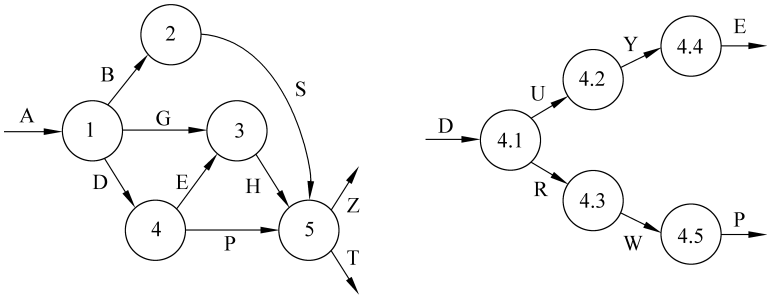
(2) 子图中加工的编号由子图号、小数点、局部顺序号连接而成。

注意：顶层图不必编号，加工一般只有一个；其下一层编号为0，该层数据流图中加工的编号就是0.1，0.2，0.3，…，通常省略小数点和前面的0，所以这些加工的编号就是1，2，3，…。

一套分层的数据流图可按编号次序用活页形式装订起来，形成一本便于查阅的资料。

2. 父图和子图的平衡

图5-9是父图和它的一张子图。父图中的加工4被分解成子图中的5个加工，子图是对父图中的加工4的描述，差别仅在于子图是详细的描述而父图是抽象的描述而已，所以子图的输入、输出数据流应该同父图中加工4的输入、输出数据流完全一致。

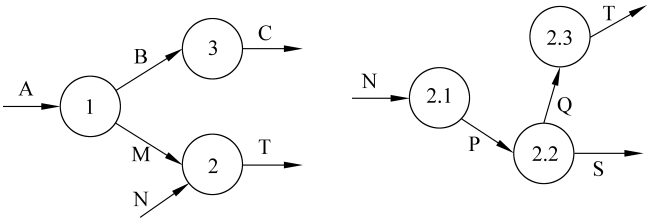


(a) 父图　　(b) 子图

图5-9　平衡的父图和子图

父图中某个加工的输入和输出数据流应该同相应的子图的输入和输出数据流相同，分层数据流图的这种特点称为平衡。更具体地说，平衡是指在借助数据词典并可忽略枝节性的数据流的情况下，子图的所有输入和输出数据流必须是其父图中相应加工的输入和输出数据流。图5-9中的父图和子图是平衡的，因为父图中加工4的输入和输出数据流与子图中的输入和输出数据流完全相同。图5-10中的父图和子图是不平衡的，因为子图中没有输入数据流与父图中加工2的输入数据流M相对应，另外，子图的输出数据流S在父图中也没有出现。

父图和子图必须平衡，这是分层数据流图的重要性质。平衡的分层数据流图是可读、可理解的；反之，如果父图和子图不平衡，这套数据流图就无法理解。

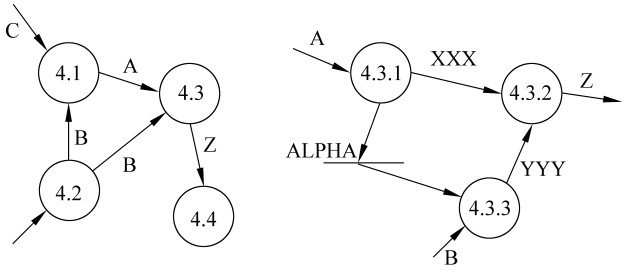


(a) 父图　　(b) 子图

图 5-10　不平衡的父图和子图

3. 局部文件

图 5-11 的两张数据流图是平衡的,但是子图中的文件 ALPHA 为什么在父图中没有画出呢? 这是因为 ALPHA 是完全局部于加工 4.3 的,它并不是父图中各个加工之间的交界面,根据"抽象"原则,在画父图时,只需画出加工和加工之间的联系,而不必画出各个加工内部的细节,所以父图中不必画出文件 ALPHA,同理,子图中的数据流 XXX、YYY 也不必画出。



(a) 父图　　(b) 子图

图 5-11　局部文件的使用

文件到哪一层才画出来呢? 原则是: 当文件被用作数据流图中某些加工之间的交界面时就要画出。

4. 分解度

分解一个系统的最终目的是将其分解到可以用只包含基本加工的数据流图来表示。分解有两个方法: 一个是直接画出一张只包含基本加工的数据流图;另一个是逐层分解。分解其实质就是如何将一个加工分解为子加工的问题,层次过多,会给理解带来困难。这就要求在分解时遵循以下原则:

(1) 分解在逻辑上应合理、自然,不能作硬性分割。

(2) 在保证数据流的易理解性的前提下,尽量使分解层次少,也就是在同一层中可以适当多分解成几部分。

(3) 分解要均匀。即在一张数据流图中,不要出现一些加工已是基本加工,另一些加工还要分解好几层(如三四层)的情况。绝对均匀不可能,但不要相差太大。

(4) 一个加工最多分解为 7 个子加工。

(5) 上层可分解得快一些,下层应分解得慢一些。

5.2.5 数据流图的改进

对一个问题、一个系统的理解不可能一下子就十分完整、全面。当初步把数据流图画完之后，可能会发现一些不当之处，要把它们纠正过来。另外，为了今后工作的顺利开展，应提高数据流图的完备性，这就涉及对数据流图正确性的检查和改进，以提高它的易读性。

1. 检查数据流图的正确性

一般可以从以下3个方面来检查数据流图的正确性：数据守恒(数据平衡)、文件的使用、父图和子图的平衡。

1) 数据守恒

数据不守恒有两种情况。

(1) 某个加工只有输出而没有输入，即一个加工用以产生输出的数据并没有输入给这个加工，这时肯定有数据流漏画了。例如，在图5-12中，“决定比赛名单”这个加工根据“项目”和“运动员名单”产生“项目参加者”。如果“运动员名单”和“项目参加者”组成如下：

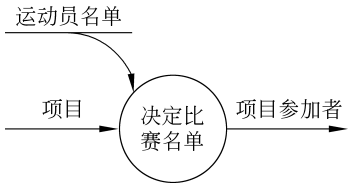


图5-12 有输出而无输入

运动员名单＝队名＋姓名＋项目

项目参加者＝项目＋姓名＋运动员号

可以发现，这个加工要输出“运动员号”这个数据，但是并没有接收到它，所以一定有数据流被遗漏了。

(2) 某个加工的某些输入没有从这个加工输出。对这种情况，要考虑这些输入是否必要，若不必要，应将多余的输入数据流去掉。

2) 文件的使用

在画数据流图时，应该注意加工与文件间数据流的方向。加工要读文件，则数据流的箭头是指向加工的；加工要写文件，则数据流的箭头是指向文件的。如果加工要修改文件，则数据流的箭头也应指向文件而不是双向的。只有当加工除了修改文件之外，为了其他目的，还要读该文件时，数据流才是双向箭头。应当注意，一般有写文件的加工，同时就会有读文件的加工，否则一定是某些加工漏画了。同时还应正确地画出加工与文件之间数据流的方向。

3) 父图和子图的平衡

父图和子图的数据流不平衡是一种常见的错误。不平衡的分层图是无法使人理解的，因此应当检查父图与子图的输入与输出的一致性。

2. 提高数据流图的易读性

提高数据流图的易读性可以从以下3个方面进行：简化加工之间的联系、分解均匀、适当命名。下面分别讨论。

1) 简化加工之间的联系

合理的分解应是将一个问题分成相对独立的几个部分，这样，每个部分就可单独理解。要增强各个加工的独立性，就必须使它们之间的联系少，也就是加工间的数据流线少。为此，应尽量减少加工间输入和输出数据流的数目，不画多余的数据流线(否则有重复的信息)。

2) 分解均匀

理想的分解是将一个问题分解成大小均匀的几个部分。当然这点是不易做到的,但是应该避免特别不均匀的分解。如果在一张数据流图中,一些加工已是基本加工,而另一些却还可进一步分解三四层,这张数据流图是不易被理解的,因为其中某些部分描述的是细节,而另一些部分描述的却是较高层的抽象。遇到这种情况,也应考虑重新分解。

3) 合理命名

数据流图中各成分的命名与数据流图的易理解性直接有关,所以应该注意命名要确切、无二义性、不含糊。

3. 再分解

发现前面的分解有问题,就需要重新分解。例如,当把数据流图分解到某一加工的子图时,发现子图是由若干个相对独立的部分组成的,而在父图中却是在一个加工中完成的,那么就需重新分解父图中的相应加工,如图 5-13 和图 5-14 所示,这里只是一种简单的情况。

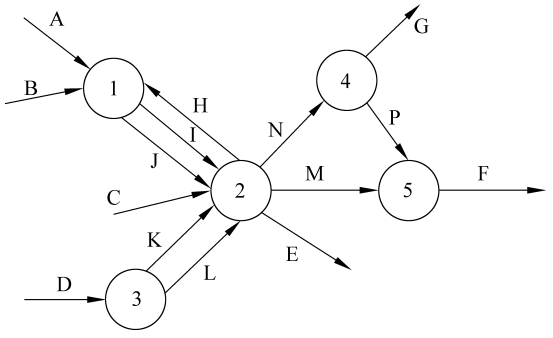


图 5-13　父图

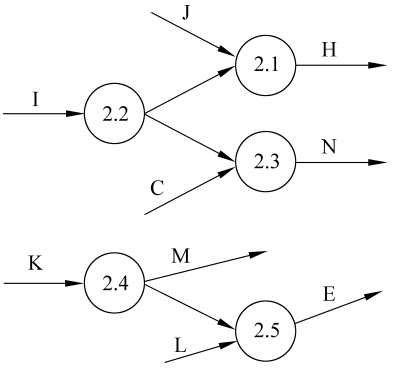


图 5-14　子图

当出现要重新分解整个父图时,可以按如下步骤进行。

(1) 把需要重新分解的某张图的所有子图连接成一张数据流图。

(2) 把子图分成几部分,使各部分之间的联系最少。也就是说,将某个加工放入哪一部分,应根据使各部分之间的联系最少的原则来决定。这当然既是这一步的目的,也是重新分解父图的目的之一,因为这样交界面清楚,有利于以后的各阶段工作。

(3) 重新建立父图。即把第(1)步所得的每一部分画成一个圆(将相应部分抽象为一个加工),而各部分之间的联系就是加工之间的交界面了,如图 5-15 所示。

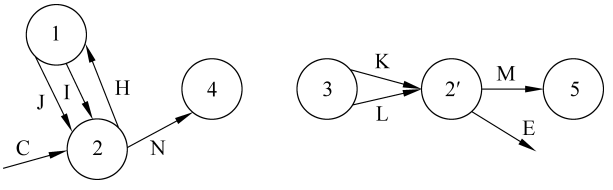


图 5-15　重新建立的父图

(4) 重新建立各张子图。因为在第(2)步中已将原有子图的设计打乱了,这时只需把第(2)步所得的图按各部分的界面剪开,然后即可进行子图重建,如图 5-16 所示。